Name of Faculty: Alka Gulati

Designation: Associate Professor

Department: MCA

Subject: Data Structure

Unit: III

Topic: Expression Tree

Expression Tree:-

An expression tree is a tree built up from the simple operands as the leaves of binary tree and operators as the non -leaves of binary tree. It is a special kind of binary tree in which:-

(i) Each leaf node contains single operand.

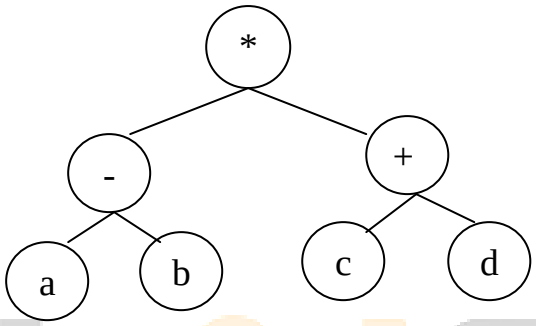
(ii) Each non-leaf node contains a single binary operator.

(iii) The left and right subtrees of an operator node represent sub-expression that must be evaluated before applying the operator at the root of the subtree.

The levels in a binary expression tree represent the precedence of operators. The operators at the lower level must be evaluated first and

then the operators at the next level and so on and at the last operator at the root node is applied and there by the expression is evaluated.

Eg. Expression: - (a-b) * (c+ d)



Prefix expression is - *-ab+cd similar to preorder traversal

Postfix expression is : - ab-cd+* similar to postorder traversal

Construction of an Expression Tree:-

(a) From prefix and infix expression

1. Read the prefix expression, the first element becomes the root.
2. Now scan the infix expression till you get an element found in step1. Place all the elements left of this element to the left of it and other element to right of it.
3. Repeat steps 1 and 2 till all the elements from infix expression gets placed in tree.

4. stop

(b) From infix and postfix expression.

1. Read the postfix expression, the last element becomes the root.
2. Now scan the infix expression till you get an element found in step1. Place all the elements left of this element to the left of it and other element to right of it.
3. Repeat steps 1 and 2 till all the elements from infix expression get placed in tree.
4. Stop

(c) From a given postfix expression.

1. Read the element from postfix expression from left to right.
2. If it is an operand then push address of this node onto stack go to step 1.
3. If the element is an operator then pop twice which gives the address of the two operands. Create a new node for this operator and attach the operand to the left and right branch. Push the address of this node onto stack .
4. If all the elements from postfix expression are not read then go to step 1.

5. Pop the contents of stack which give the root address of expression tree.

6. Stop

Eg. Construct expression tree from following expressions:-

Postfix expression:- 5,3,2,^,*,3,7,3,+,10,/,+/

Infix expression :- (5*3^2)/(3+(7+3)/10)

Sol. From postfix we get root i.e. / ,hence (5*3^2) is left subtree and (3+(7+3)/10) is right subtree. Now again find root from postfix for right You get + , now 3 is left and is (7+3)/10 is right, then 3 is left child.

In right subtree / is root , (7+3) is left subtree and 10 is right subtree.

10 is root for right subtree. Now in left subtree + is root and 7is left child and 3 is right child.

Similarly for left subtree * is root and 5 is left subtree and 3^2 is right subtree.

Now again from right subtree root is ^, 3 is left child and 2 is right child.

The tree is constructed as follows:-